Notes on Latent Semantic Analysis

Costas Boulis

1 Introduction

One of the most fundamental problems of information retrieval (IR) is to find all documents (and nothing but those) that are semantically close to a given query. Early approaches in IR used exact keyword matching techniques to identify relevant documents with unsatisfactory results. There are two main reasons why these approaches fail:

- *Synonymy.* Two different words may refer to the same concept. For example *car*, *automobile*. If a document contains just the word *car* and the query just the word *automobile* then exact keyword matching techniques will fail to retrieve this document.
- *Polysemy.* The same word may refer to different concepts, depending on the context. For example the word bank (river bank or money bank). Context is not a simple matter of N-grams but can extend far in a document.

This many-to-many mapping between words and concepts (or semantics) makes it difficult, if not impossible, to handcraft rules. For example some early approaches used a hand-crafted dictionary where synonyms where specified. Each query was then expanded with many synonyms of each word appearing in the query and an exact keyword match was then performed. However such approaches don't handle the issue of polysemy and will return a high number of irrelevant documents.

One way to model the interaction between words and concepts is to view a query as a noisy observation of a concept. We can think of a concept as being associated with a probability distribution function (pdf) over the vocabulary words. For example a topic about *car maintenance* will have a higher probability than other topics for the words *car*, *mechanic*, *parts* and so on . But when a query is formed, not all these words with their true frequencies will appear. Some words that are highly associated with a certain topic may even not appear at all. Under this perspective a query is a noisy observation of an original topic (or perhaps a set of topics).

2 Principle Components Analysis

Principle Components Analysis (PCA) is frequently used as a data compression technique and works by removing redundancies between dimensions of the data. There are many references to PCA in the literature, here I will just briefly mention some of its main characteristics without using a lot of mathematical notation.

PCA projects the original data in a lower dimensionality space such that the variance of the data is best maintained. Let's suppose that we have N samples each of dimension M. We can represent these data as a matrix X of $N \times M$ size. It is known in linear algebra that any such matrix can be decomposed in the following form (known as Singular Value Decomposition or SVD):

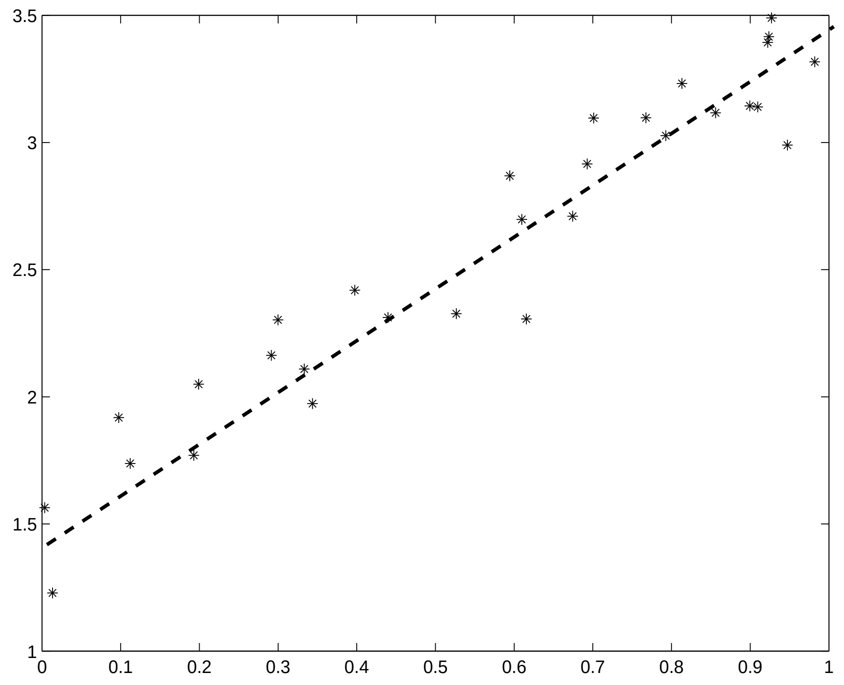

Figure 1: Projecting the 2-D data to a line

$$X_{N\times M} = U_{N\times M} S_{M\times M} V^T_{M\times M} \tag{1}$$

Where U and V are orthogonal[1] matrices of $N \times M$ and $M \times M$ size respectively. The S matrix is a diagonal $M{\times}M$ matrix (assuming $N > M$). The diagonal elements of S are the eigenvalues of X and the matrices U and V contain the eigenvectors of X[2]

If we order the eigenvalues according to their magnitude and keep the highest m of them then we essentially map the original M dimensional space to a m dimensional space with the least distortion, where distortion is defined to be the mean square error between the original and mapped points. So now we can write:

$$X \approx \hat{X}_{N\times m} = U_{N\times m} S_{m\times m} V^T_{m\times M} \tag{2}$$

In Figure 1 an example for 2-D data and their minimum mean square error projection to a single dimension, is shown.

We can observe that from all the lines that exist, the one where the projected data would have a variance as close as possible to the original data is the one shown.

We can now see why PCA is a compression technique. Before we would need 2 real numbers to store each one of the data points, now we need 1 (with some distortion associated with it).

Even if we don't project to a lower dimensionality space (if $m = M$) PCA may still be useful since it decorrelates[3] the dimensions. This is shown in Figure 2:

Such a transformation can be useful for many feature normalization tasks in practice.

[1]A $N{\times}M$ matrix U is defined to be orthogonal iff $U^TU = I$, i.e. its inverse is the same as its transpose

[2]If X is a $N{\times}M$ matrix then u and λ are respectively an eigenvector and eigenvalue of X iff $Xu = \lambda u$. The eigenvalues are scalar and the eigenvectors are vectors of length $min(N, M)$

[3]Two random variables Y and Z are defined to be correlated if the knowledge of one can help us know the other through a *linear* relation

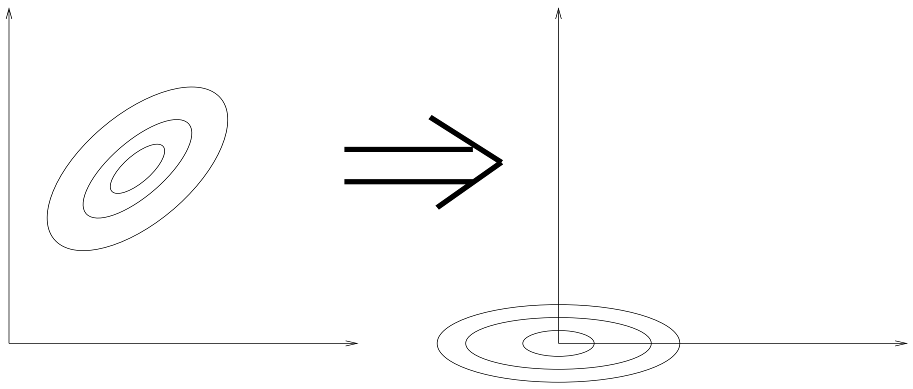

Figure 2: PCA as a decrorrelating transform

But PCA is not just useful for compression. We can see this method as a way to estimate the underlying relation between different variables. For example in Figure 1 we can think that the x axis is gas gallons and the y axis is miles I get on a specific car, road, environment and driving conditions e.t.c. We know that there is a linear relation between these two variables but due to noisy sensors and imperfect road conditions the actual measured points are as shown in Figure 1. Thus PCA is a way to remove the noise from the data and uncover the "true" relationship between different variables.

3 Factor Analysis

PCA is a method in linear algebra and as any method in linear algebra all the variables are deterministic. Factor Analysis (FA) is a similar method but with probabilistic foundations. In FA we assume that the observed data x are generated as a linear combination of a (usually small) set of hidden factors[4].

$$x = \Lambda f + \epsilon \tag{3}$$

where f is a vector of size m, Λ is a $M \times m$ vector, x is a vector of size M and ϵ is a random vector of size M. The $\epsilon_i, i = 1..M$ are uncorrelated and x is uncorrelated of ϵ given f.

During training, the task is to estimate the common hidden factors given a set of observed vectors x. During testing, the task is to find the projections of a test vector x to the already estimated hidden factors f.

If x is the original M-dimensional vector then from Equation 3 we can write:

$$p(x) \approx \sum_f p(x|f)p(f) \tag{4}$$

From Equation 4 we can see that FA is an *unsupervised* way of learning the m-dimensional factors from M-dimensional data. The dimensionality reduction is important because of data sparsity issues. By using the dimensionality reduction we simplify our model by requiring fewer parameters to be estimated. And if the data are highly correlated there is no sacrifice in the modeling capacity of the projected space.

[4]This is the one-mode FA, where the factors refer to just one variable. More similar to SVD is the two-mode FA which is described in the next section.

In PCA each vector can also be decomposed as a linear combination of a set of basis vectors. The difference between PCA and FA (other than the former is a linear algebra method and the latter a probabilistic one) is that PCA preserves the variance while FA preserves the covariance. Therefore PCA is insensitive to rotation but sensitive to scaling and vice versa for FA. For an excellent discussion on PCA, FA and probabilsitic extensions of PCA (like mixtures of PCA) see [4]. Probabilistic methods have a number of advantages over algebric ones including:

- *Value of m.* An issue with both FA and PCA is what is the optimum value of m,i.e. what is the value of m that best represents the data (according to a predetermined criterion). There is a sound body of methods and theory for selecting the complexity of statistical models, while in PCA we have to rely on heuristics or simple techniques.
- *Training and updating.* For most applications where the number of training samples and dimensionality is high, as in IR, PCA will result in a huge matrix which needs to be decomposed. On the other hand FA can make use of the wealth of EM[5] variants that exist and apply techniques which are both fast and robust to estimation errors.
- *Generalization* It is well known in statistics that FA is a special case of more complex models (called Graphical Models). Under this perspective there are many ways to visualize generalizations of FA. See for example [3].

4 Back to Information Retrieval

How all these relate to Information Retrieval? First of all, we need a representation of documents. The vector-space representation is one of the most popular, where each document is represented as a vector of the words appearing in the document. Some very common words like *the, a, and* e.t.c. are removed and the remaining words are stemmed, based on the assumption that words with the same prefix refer to the same topic (for example *run, runner, running*). The dimensionality M of each one of the documents is usually very high, in the order of dozens of thousand and usually each entry of the vector is either 0 or 1, meaning that a term either appeared at least once in the document or didn't appear at all. If we have N documents available then we can construct an $N \times M$ matrix where each row corresponds to a document. We can think of this matrix as a description of both terms and documents where a row describes a certain document and a column describes a specific term.

Further assume that we have selected m and used PCA, Equation 2. The diagonal matrix S refers to the common factors. Notice that these factors are common for both terms and documents, allowing for uniform representation. This allows us to ask questions like *How similar are word W and query Q* apart from the standard *How similar are words W_1 and W_2 or documents D_1 and D_2*. The $N \times m$ matrix U projects the terms into the m-dimensional space and the $M \times m$ matrix V projects the documents into the m-dimensional space. So if d is a vector describing a document and t is a vector describing a term we have:

$$\hat{d} = V^T d \tag{5}$$

$$\hat{t} = U^T t \tag{6}$$

Notice also that although the original representation of terms or documents is discrete the projected points lie on a continuous space. We can now measure the proximity of two points by using the Euclidean measure in the m-dimensional space instead of the original M-dimensional space.

[5]Expectation-Maximization (EM): a popular algorithm in Machine Learning for unsupervised learning

At the introduction section, I mentioned that we can see each query or word as a noisy realization of a number of topics. This is mathematically described by Equation 3, where x is the observation vector f are the topics or semantics estimated on a number of documents and ϵ is the error or noise.

We can have the two-mode FA for documents d and terms t by using:

$$p(d,t) \approx \sum_f p(d|f)p(t|f)p(f) \tag{7}$$

Notice that Equation 7 is completely symmetric in both the documents and the terms. Here f are the common factors or topics associated with a collection of documents or terms. Perhaps the strongest advantage of FA and PCA is that the procedure of uncovering the hidden or latent semantics is totally unsupervised.

Latent Semantic Analysis (LSA) as presented in [1], is exactly the same as two-mode PCA or SVD and Probabilistic LSA (PLSA) as presented in [2], is the two-mode extension of probabilistic PCA as presented in [4].

How LSA and its variants handle the two fundamental problems of IR, namely synonymy and polysemy? Synonymy is captured by the dimensionality reduction of PCA. Words which are highly correlated will be mapped in close proximity in the low dimensional space. Polysemy, on the other hand cannot be handled by PCA because inherently the same transform is applied on all the data. For example suppose that we have a collection of documents and at the first half the topic is about fishing so there is correlation between the words *river* and *bank* and at the second half the topic is about the economy and now the words *money* and *bank* are correlated. If PCA is applied in this scenario it will map in the same direction the words *river*, *bank* and *money*. The work in [2] offers a way to resolve polysemous words by calculating the quantity $p(w|f)$ (see Figure 3 from Hofmann's paper). A better way would be to estimate a mixture of probabilistic PCA, so that we learn jointly the transformations and the class parameters.

5 Handling speech recognizer output

All the variants of LSA developed so far are for text taken from newspapers, technical documents, web pages e.t.c. This means that the form processed thus far is from lexically correct sources[6]. But what about the case where the text is the output of automatic speech recognizers. The outputs will have errors in many levels and the challenge now is how to uncover semantics from erroneous sentences. We can think of this problem as estimating a double-hidden parameter and can be represented in a nice way through Graphical Models. There are additional issues that need to be taken care of, like the fact that if a word is very likely to be erroneous then its neighbors will probably also be erroneous.

6 Summary

There are two main points to be remembered for Latent Semantic Analysis:

- LSA is an unsupervised way of uncovering synonyms in a collection of documents.
- Maps both documents and terms to the same space, allowing queries across variables (similarity of terms and documents).

[6]Note that we don't care about grammaticaly and syntactically correct sentences because LSA assumes the bag-of-words model.

References

[1] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, “Indexing by Latent Semantic Analysis,” *Journal of the American Society of Information Science*, pp.391-407,1990.

[2] T. Hofmann, “Probabilistic Latent Semantic Analysis”, *Proceedings of Uncertainty in Artificial Intelligence*, 1999.

[3] M. Jordan, C. Bishop, “An Introduction to Graphical Models”.

[4] M. E. Tipping , C. M. Bishop, “Mixtures of Probabilistic Principal Component Analysers”, *Neural Computation*, vol. 11, no. 2, pp. 443-482, 1999.